A Correlated Strategic Guide for Software Testing

Christopher L. Harlow
The George Washington University Law School

Dr. Santa Falcone
University of New Mexico

Due to the complexity of software development, completed programs are rarely ever flawless. Instead, they exist in a state of constant refinement. As a result, large-scale customized software programs are routinely purchased and employed while still experiencing significant problems. Because of the programs' scale, these problems cost public- and private-sector organizations billions of dollars each year in productivity losses and repair expenses. In the study in this article, test and field data from a large-scale software development project were analyzed using Chi-square goodness-of-fit tests, p-tests, and binary logit regression. The results of this series of calculations support using initial test deficiencies in mid-production software tests to guide later iterative testing to increase deficiency exposure. When used during software development, this strategic test guide is expected to improve testing, expose problems earlier, help lead to higher quality end-products, and ultimately reduce the large losses organizations experience due to customized software defects.

Information technology advances have equipped the U.S. military with an extraordinary arsenal of weaponry and supporting materiel. To keep the arsenal current, the Department of Defense (DoD) modified its acquisition strategy to reflect the iterative nature of information technology (IT) development. Both public- and private-sector large-scale software-intensive acquisitions use iterative strategies to conduct the *try-before-you-buy* testing of these purchases. This article briefly reviews literature on software testing and describes an iterative test strategy used in an actual large-scale military software acquisition.

The Timing of Software Testing

In the past, a series of increasingly stringent development tests controlled quality and guided DoD large-scale customized IT acquisitions through the procurement process. Right before acquisition fielding, an operationally realistic test evaluated the product's functional effectiveness within its intended environment [1]. Thus, multimillion-dollar purchase decisions depended largely upon successful operational tests at the end of production.

In the United States in 2002, the National Institute of Standards and Technology calculated the annual cost of these operational test failures in the U.S. public and private sectors at $59.5 billion [2]. An independent study found that more than half of IT acquisitions doubled their initial budget and schedule projections, the average acquisition provided only 61 percent of the desired functionality, and one-third of software-intensive projects were ultimately cancelled [3].

In IT acquisitions, the desired software end-product is seldom clearly defined [4]. The identification of requirements and the development of the software to fulfill them are progressive and often become intertwined, fueling a spiral in which unanticipated requirements emerge [5]. Government agencies have referred to this process as *evolutionary acquisition* and *spiral development*. Their civilian counterparts use the phrases *agile development* and *extreme programming* [6].

In the recent past, government testing was insensitive to this process of software development. In 2000, the U.S. General Accounting Office (GAO) harshly criticized DoD evaluation methodologies and concluded that late operational test failures consistently plagued DoD purchases, particularly software-intensive systems [7]. Accordingly, the GAO denounced the traditional practice of employing testing as a *watershed event* in sole support of fielding decisions. The failure of traditional practices combined with the technological revolution led directly to the development of alternative test strategies, including the method described in this article.

After studying government and private practices, the GAO concluded that the most effective method of exposing deficiencies was iterative operational testing. GAO further recommended the philosophy developed by AT&T: *Break it big early* [7]. According to AT&T, the earlier a problem is discovered, the easier and less expensive it is to fix, making software development more cost-effective. Other benefits include encouraging technological exploration and advancement, controlling risk [8], exposing unanticipated requirements, and enabling their eventual fulfillment.

Spurred by success, incremental testing has become the industry standard for software development projects. For example, during the acquisition of the Theater Battle Management Core System, evaluators successfully mitigated risk by operationally testing each basic system component prior to fielding. The test team also observed that the incremental strategy facilitated requirements development as operators became increasingly familiar with the system [9]. The DoD employed the strategy with several programs, significantly improving its acquisition and testing practices.

In addition to incremental testing, there is also near unanimous agreement about using operationally realistic testing early in the development cycle. Past and present DoD Operational Test Agency leaders assert that early operational tests streamline the production process, improve requirements definition, and provide valuable feedback [10, 11]. The Global Command and Control System test team, one of the first DoD programs to effectively use it, found that early operational testing increased operator system familiarity, which, in turn, increased the number of deficiencies exposed during testing [12].

In March 2002, the Giga Information Group estimated that two-thirds of private-sector software projects would employ *agile development* within the next two years. Using terms nearly identical to those in the government lexicon, the group identified these projects as those that are divided into smaller phases and require more frequent testing [6]. According to the prevailing opinion in

the public and private sectors, the best response to the challenges of IT testing is incremental operationally oriented evaluations beginning in the early stages of system development.

Early and Continuous Software Testing

While the community has unanimously embraced the need for earlier testing, there is no consensus on how to conduct early testing. Experts have proposed merging developmental and operational tests, instituting sustained independent operational testing, using Bayesian hierarchical models [13], and replacing operational testing with mathematical models. In addition, some argue that those who will actually operate the completed system should be active observers during early developmental tests [3].

Ideally, involving primary users in early testing expedites identification, prioritization, and resolution of exposed deficiencies, and results in lower costs due to reduced operational testing [4]. This modified early testing with primary users still culminates in a final operational test. Thus, while using operators as testers and separating large-scale software acquisitions into incremental segments conform to the new paradigm, this modified testing also retains the traditional concept of a final operational test of the whole system.

Several experts eschew the implicit requirement for physical testing and suggest the utilization of risk assessment models. Thompson proposes four strategies ranging from the traditional production model to various levels of operationally realistic testing, determined by a variety of statistical inputs [14]. Expanding on Thompson's proposal, Arnold's mathematical model would effectively eliminate the need for detailed testing of specific aspects of a potential acquisition, ostensibly saving substantial resources [15]. While the substitution of mathematical modeling for testing is not entirely accepted within the community, proponents of mathematical modeling assert that accurate models can be built. Systematic refinement of mathematical models is anticipated to improve their predictive abilities [16].

The model or strategy proposed in this article employs Thompson's idea of using statistical data in testing. However, the use of statistical data here will identify specific test activities, not just test levels. It also will provide a guide for the iterative incremental testing process rather than an outright replacement of test activities.

Testing Using Correlation

The DoD commissioned a study in May 2000 to evaluate the accuracy of operational tests by comparing test results of 11 systems to wartime field data [17]. The limitations of this study were as follows: Low-level task-based test data was compared with strategic-level operational data; a standardized deficiency tracking method before and after fielding was not used; all 11 systems had been significantly altered after testing and many of the modifications were not documented; and, finally, there was an extremely high turnover rate of test personnel. Turnover is not isolated to the military: Civilian test teams often encounter turnover rates as high as 80 percent in test periods only a few months in length [18]. To credibly compare data from one iterative test to another, the identification of test and field deficiencies should be standardized and the only substantive system changes should be fixes to deficiencies exposed during official testing.

The closest approximation to this article's correlation model was the incremental development and testing of IBM's integrated results database for the 2000 Summer Olympics. IBM utilized standard Orthogonal Defect Classification to categorize system defects along an x-y graph, revealing defect similarities and trends the IBM team termed *triggers*. The IBM team assumed that a high trigger incidence rate revealed an area of weak code and "used the insight to guide test teams toward more effective defect discovery" [18]. The IBM trigger metrics represent an initial attempt by the software industry to relate test data to future performance in a manner aimed at improving the production process.

As noted earlier, the specific requirements of large-scale software programs are unclear at the inception of the design and production process. The correlation of iterative test results is proposed here to enable discovery, more specific clarification, and fulfill end-user needs during production and testing. This approach also helps manage the eight factors that influence success or failure during software testing: production, infrastructure, training, personnel, communications, operations, maintenance, and environment [1].

Regarding the first factor, production, according to software production expert M.S. Phadke, faulty coding tends to be regional and hence predictable [19]. Regional, as it is used here, refers to either a section of the code or a physical location where the code is generated. As per Phadke's observation, there will be regions of computer code that are highly correlated with deficiencies in the initial stages of design and production (prior to the first test). These regions should be more closely observed in the first test for the specific types of deficiencies identified during production. Any region with a large number of deficiencies during the first test would then be targeted for emphasis in the next production cycle and for extensive testing during the second test. Comparing by region the deficiencies exposed during the first test to those exposed during the second test would again target the regions that should receive more emphasis in the following production cycle and extensive testing during the third test, and so on, iteratively encouraging the optimal identification of unexposed deficiencies.

Comparing this approach to IBM's, the triggers identify only the regions of code with problems. They do not indicate the relative significance of the difference between the regions and do not track the eight factors. In this approach, data about the eight factors was collected during two tests of a large-scale software program. This enabled analysis of the deficiencies exposed and each of the eight factors. This analysis identified the regions of code that were problematic, the difference between regions, and which of the eight factors were significantly related to the exposed deficiencies.

Methodology

The data for this study came from two tests (TEST1 and FIELD) of the large-scale software program Deliberate and Crisis Action Planning and Execution System (DCAPES), designed to serve an information technology need of the U.S. Air Force [20]. For TEST1, operator information and the results of the testing of 53 tasks within the first increment of DCAPES were collected from five worldwide locations during two weeks in June of 2001. The 53 specific tasks were organized into the following six categories: operations, logistics, information analysis, reference file access, manpower data management, and system administration [20]. Fifty-two operators assessed the system's capabilities and performance in 4,592 discrete actions in a simulated operational environment. Data collected in TEST1 included the overall recording of the outcomes of the test, deficiency documentation, operator information,

and operator satisfaction surveys.

The first increment of DCAPES was then fielded in March 2002. FIELD, the second data collection effort, involved recording deficiencies that were found in the normal operation of the first increment of DCAPES after it was fielded. Normal operations consisted of an estimated 225,000 discrete actions by approximately 2,000 authorized operators in 16 worldwide locations over a 10-month period. Data collected in FIELD included deficiency documentation and operator information.

Performance failures or deficiencies were documented using the exact same procedures and codes for both TEST1 and FIELD. Operators documented deficiencies with standardized forms extensively describing the problem and the circumstances leading to its exposure. While documenting the problem, the operator assigned a priority describing the impact of the deficiency on the organization's mission on a five-point numeric scale. A priority one deficiency described a catastrophic mission failure, while a priority five described an easily circumvented minor inconvenience.

After the operator assigned an initial priority, a Deficiency Review Board (DRB) convened to review the legitimacy of the problem and the appropriateness of the assigned priority. The DRB consisted of representatives from the software developer company, the operating community, and an independent government evaluator as chairman ensuring impartial review. TEST1 and FIELD both followed these same procedures, utilizing identical definitions and DRB members. This replication ensured comparable treatment of test and field deficiencies. The DRB also rejected duplicate deficiencies, ensuring each problem was documented only once. After the DRB assigned a final priority, the problem was officially documented in another database that consolidated and segregated test and field deficiencies, making this information readily available to authorized personnel.

A series of χ^2 goodness-of-fit tests, p-tests, and binary logistic regression equations were conducted on the data collected to answer the problem statement, *"Can the correlation of software test and field data be used to guide testing to increase deficiency exposure?"* The analyses are organized and discussed in three sections: system functionality, organizational trends, and operator data. System functionality refers to the performance of the DCAPES system. Organizational trends refer to how organizations impact the performance of the DCAPES system. Operator data refers to how operators impact the performance of the DCAPES system.

Findings

Regarding system functionality, this study began with the assumption that coding errors tend to be regional. One implication of this assumption is that defects are interrelated and may be predictable. Analysis of the results of the testing of the 53 system tasks within the six functional categories supports this assumption. The data indicates that tasks and categories with high execution quantities had more field deficiencies. These tasks and categories were more complex, containing a broader range of functions made possible through additional lines of code. Due to this complexity, these areas were more susceptible to errors. The binary logistic regression results also indicate a significant relationship between the execution volume of complex code and incidence of errors. Applying this finding, a test director could increase the testing of regions with high volume execution.

Another affirmation of the regionality of errors was evident in that regions of code plagued by failures during the first test exhibited additional defects in the field test. The time and resource constraints placed on testing prohibit exposure of every deficiency in any single round of testing. As a result, it is not surprising that defects remain undiscovered even in areas well tested in previous iterations. Again, a test director could alter test activities to emphasize regions with substantial failures in past tests.

An organization-wide trend that had significant impact was the adequacy of training provided to operators. The findings indicate that the categories in which operators had inadequate training had higher field deficiency quantities. Intuitively, this results from the inability of operators to stringently test these categories in the first test. This study found that DCAPES training problems were more related to the categories of tasks than the geographic location of the task execution. This provides valuable information because training courses are typically organized along both lines with category specialists trained as a group at each location.

To improve a test in progress before its completion, the awareness that operators have inadequate training in specific categories should be used to redirect testing to emphasize the reported categories using only well-trained operators. However, in an incremental development project, this feedback concerning categories in which operators are reporting significant training problems could be provided to instructors and the training adjusted to prevent inadequate training from negatively impacting later tests.

The operator data identified significant factors to guide the testing process and factors that are not as important as conventional wisdom suggests. For example, prevailing opinion is that defect exposure detracts from test execution. A variety of calculations revealed that the expected negative relationship between execution and deficiency submission did not occur. Instead, the significant relationship in this regard was that operators who submitted false deficiencies (false deficiencies are those withdrawn by the submitter or rejected by the DRB) exposed significantly fewer legitimate defects overall. This calls into question the quality of testing accomplished by these operators and suggests the participation of operators who submit false deficiencies in past tests should be curtailed in future tests.

Current conventional wisdom also asserts that test planning should entail the meticulous creation of a simulated environment that closely approximates the operational setting. The findings of this study, however, indicate that some facets of this environment may not be necessary to simulate so meticulously. For example, significant resources are devoted to recruiting representative operators; however, the findings here were that field deficiency exposure rates increase with operator skill level and system experience. Using operators in the testing process with great skill level and system experience apparently would be a more effective and efficient way to expose deficiencies than ensuring that operators representing the full available range of skill levels participate in the test.

Finally, tasks with lower operator satisfaction levels were found to contain unexposed deficiencies. These operator satisfaction results provide information about system performance that can be used to guide testing. Within the time period of one test, operators often rotate in and out. Using this to advantage, a test director could steer subsequent testing toward areas receiving lower marks on satisfaction ratings of operators rotating out. In a spiral development effort, test directors could also schedule later tests to emphasize areas receiving lower operator satisfaction ratings in previous iterations.

Conclusion

The statistical analysis within this study has specific application to the remaining tests in the development of DCAPES and the development of similar large-scale systems. Through publishing this article, information about the approach suggested here will be incorporated into the body of information about software testing. Then, as the methods here are more widely employed, they will refine testing and help improve end-product quality.

As mentioned earlier, the National Institute of Standards and Technology estimates software defects cost the United States nearly $60 billion annually. The Institute also estimates that practical approaches to software development and testing could reduce this figure by 37 percent [2]. Essentially, $20 billion can be saved each year through the implementation of relatively simple software production models. The correlation model proposed in this article is an initial attempt at capturing a small percentage of these funds. While not the most statistically elegant method, it is a method that most professionals in the work world will be able to understand and, most importantly, use. Public- and private-sector large-scale software development projects may both realize tremendous gains from a statistical correlation model.

The data analysis referred to in this article is available upon request.◆

References

1. McGowen, D.J. "C4I Operational Test and Evaluation in Support of Evolutionary Acquisition Strategy." ITEA Journal of Test and Evaluation 21.2 (2000): 34-39.
2. RTI. "Planning Report 02-3: The Economic Impacts of Inadequate Infrastructure for Software Testing." Washington, D.C.: National Institute of Standards and Technology, May 2002 <www.nist.gov/director/prog-ofc/report 02-3.pdf>.
3. Maybury, M., A. King, and J. Brooks. "Software Intensive System Acquisition – Best Practices." 2003 Acquisition Conference, 28-30 Jan. 2003, Arlington, VA <www.sei.cmu.edu/products/events/acquisition/2003-presentations/maybury.pdf>.
4. Assi, S., and M. Thompson. "Alternative Test and Evaluation Strategies for Command and Control Systems." ITEA Journal of Test and Evaluation 20.2 (1999): 21-25.
5. Axiotis, G. "Evolutionary Acquisition and Operational Testing, Time for a New Approach." Evolution 99 (1999): 1-5.
6. Sliwa, C. "Users Warm Up to Agile Programming." Computerworld 18 Mar. 2002 <www.computerworld.com>.
7. General Accounting Office. "Best Practices: A More Constructive Test Approach Is Key to Better Weapon System Outcomes." Washington: GAO, July 2000.
8. Cast, M. "Military Links Developmental and Operational Testing to Meet Technology Challenges of the 21st Century." Program Manager July-Aug. 2000: 16-18.
9. Zyroll, T., A. Johnson, and B. Connally. "Government Testing Philosophy Redefined for Theater Battle Management Core Systems." ITEA Journal of Test and Evaluation 19.1: 25-47.
10. Whittmeyer, J. "Meet DoD's Top Advisor on Operational Test and Evaluation." Program Manager May-June, 1996: 2-8.
11. Besal, R.E., and S.K. Whitehead. "Operational Testing: Redefining Industry Role." National Defense Magazine Sept. 2001 <www.nationaldefensemagazine.org/archives.htm>.
12. Bailey, M., and M. Spencer. "Engineering the Largest C4I Operational Test in Naval History." ITEA Journal of Test and Evaluation 20.1 (1999): 26-33.
13. Graves, T. "Hierarchical Models for Software Testing and Reliability Modeling." 2003 Quality and Productivity Research Conference, IBM T.J. Watson Research Ctr., Yorktown Heights, N.Y., May 21-23, 2003.
14. Thompson, M., and E. Montagne. "Using Risk Assessments to Determine the Scope of Operational Tests for Software-Intensive System Increments." ITEA Journal of Test and Evaluation 19.1 (Jan. 1988): 42-47.
15. Arnold, A.G., and W.F. Kujawa. "Test and Evaluation of Complex System of Systems." ITEA Journal of Test and Evaluation 20.3 (Mar. 1999): 33-36.
16. O'Bryon, J.F. "Meet MASTER-Modeling and Simulation Test and Evaluation Reform. ITEA Journal of Test and Evaluation 20.4 (Apr.1999).
17. Brown, S.O., and K.E. Murphy. "Air War Over Bosnia: OT&E Impact on USAF Systems Used Over Serbia." Kirtland Air Force Base, NM: Air Force Operational and Test Evaluation Center, 2000.
18. Bassin, K, and S. Biyani.. "Metrics to Evaluate Vendor-Developed Software-Based on Test Case Execution Results." IBM Systems Journal 41.1 (2002): 13-30 <www.research.ibm.com/journal>.
19. Phadke, M.S. "Planning Efficient Software Tests." CROSSTALK Oct. 1997 <www.stsc.hill.af.mil/crosstalk/1997/10/index.html>.
20. Harlow, C. "DCAPES Operational Assessment Final Report." Kirtland Air Force Base, NM: Air Force Operational and Test Evaluation Center, 2001.

About the Authors

Christopher L. Harlow currently attends The George Washington University Law School. Previously, he was a test manager for three years at the Air Force Operational Test and Evaluation Center in New Mexico where he designed and executed worldwide operational tests of The Deliberate Crisis Action Planning and Execution System. Harlow has a Bachelor of Science in economics from the U.S. Air Force Academy and a Master of Public Administration from the University of New Mexico.

The George Washington University Law School
E-mail: charlow@law.gwu.edu

Santa Falcone, Ph.D., is an associate professor in the School of Public Administration at the University of New Mexico and teaches public budgeting and public finance.

University of New Mexico
2085 Anderson School of Management
Albuquerque, NM 87131
Phone: (505) 277-4934
Fax: (505) 277-7108
E-mail: falcone@unm.edu